21-10-00xx-00-21a.doc

Project	**IEEE 802.21 Media Independent Handover Services** **IEEE 802.21a: Security** **<http://www.ieee802.org/21/>**
Title	**Draft IEEE 802.21a Document Proposal**
Date Submitted	**July 15, 2010**
Source(s)	**Lily Chen (NIST)**
Re:	IEEE 802.21 Session #39 in San Diego, CA
Abstract	This document presents a document proposal for IEEE 802.21a. The draft includes the selected proposals. The purpose is to initiate a draft standard for further development.
Purpose	Task Group Discussion
Notice	This document has been prepared to assist the IEEE 802.21 Working Group. It is offered as a basis for discussion and is not binding on the contributing individual(s) or organization(s). The material in this document is subject to change in form and content after further study. The contributor(s) reserve(s) the right to add, amend or withdraw material contained herein.
Release	The contributor grants a free, irrevocable license to the IEEE to incorporate material contained in this contribution, and any modifications thereof, in the creation of an IEEE Standards publication; to copyright in the IEEE's name any IEEE Standards publication even though it may include portions of this contribution; and at the IEEE's sole discretion to permit others to reproduce in whole or in part the resulting IEEE Standards publication. The contributor also acknowledges and accepts that this contribution may be made public by IEEE 802.21.
Patent Policy	The contributor is familiar with IEEE patent policy, as outlined in Section 6.3 of the IEEE-SA Standards Board Operations Manual <http://standards.ieee.org/guides/opman/sect6.html#6.3> and in *Understanding Patent Issues During IEEE Standards Development* <http://standards.ieee.org/board/pat/guide.html>.

IEEE Standard for
Local and metropolitan area networks—

Part 21: Media Independent Handover Services

Amendment: Mechanisms to protect MIH and to enable MIH assisted proactive authentications

Abstract: This amendment specifies the extensions to IEEE Std 802.21-2008 for security mechanisms to protect media independent handover service and mechanisms to use MIH to assist proactive authentications to reduce the latency due to authentication and key establishment when handovers happening between heterogeneous access networks that support IEEE 802.21.

Keywords:

IEEE Standard for
Local and metropolitan area networks—

Part 21: Media Independent Handover Services

Amendment: Mechanisms to protect MIH and to enable MIH assisted proactive authentications

1 Overview

1.3 General

Change the following text as indicated:

The following items are not within the scope of this standard:

- Intra-technology handover [except for handovers across extended service sets (ESSs) in case of IEEE 802.11]
- Handover policy
- ~~Security mechanisms~~ Media specific protection mechanisms
- Enhancements specific to particular link-layer technologies that are required to support this standard (they will be carried out by those respective link-layer technology standards)
- Higher layer (layer 3 and above) enhancements that are required to support this standard

2 Normative references

Insert the following normative references:

IEEE Std 802.21™-2009, Information Technology—Telecommunications and information exchange between systems—Local and metropolitan area networks—Specific requi; Part 21: Media Independent Handover Services

IETF RFC 3748 (2004) Extensible Authentication Protocol (EAP)

IETF RFC 5296 (2008) EAP Extensions for EAP Re-authentication Protocol (ERP)

(Editor's note: more references to be added)

3 Definitions

Insert the following definitions:

3. 1x Media specific association protocols (handshake): Protocols defined with a specific media executed between a mobile node and a point of attachment to be connected to the network.

3. 2x Media specific protection mechanisms: An encryption, an integrity protection, or an authenticated encryption mechanism applied to media specific layer to protect data traffic.

3. 3x Media specific network access authentication: An authentication protocol for media access purpose. It may include a key establishment subroutine to establish keys to be used in media specific protection mechanisms.

3. 4x Extensible authentication protocol (EAP): A network access authentication framework specified in IETF RFC 3748. It can support different authentication methods, called EAP methods.

3. 5x Proactive authentication: A network access authentication executed prior to handover.

3. 6x Authenticator: A network entity to execute EAP with a mobile node called a peer. An authenticator can use a backend server to conduct EAP authentication

3. 7x Media specific authenticator: An authenticator used for a media specific network access authentication.

3. 8x EAP server: A centralized backend server to be used by authenticator to execute EAP.

3. 9x EAP Re-authentication: An authentication protocol defined in IETF RFC 5296, which uses an established re-authentication key through an EAP. An EAP re-authentication protocol includes fewer messages than required for an EAP protocol.

(Editor's note: more to be added.)

4 Abbreviations and acronyms

Insert the following abbreviations and acronyms:

EAP Extensible Authentication Protocol

ERP EAP Re-authentication Protocol

(Editor's note: add as needed)

5 General architecture

Insert the following text:

5.1.9 Proactive authentication and key establishment

This standard allows a mobile node to conduct a proactive authentication and key establishment before a handover to reduce the latency. The proactive authentication can be an authentication protocol specified by a network with specific media or an authentication to access media independent service. A successful proactive authentication and key establishment will allow a media specific point of attachment (PoA) obtaining a key or keys which will be used to derive session keys to protect the communication link between the MN and the PoA.

6 MIH Services

6.3 Media independent event service

6.4 Media independent command service

6.5 Media independent information service

6.5.4 Information elements

Option 1: Insert the following Information Elements in Table 10[1]:

Name of information element	Description	Data type
General information elements		
…		
Access network specific information elements		

[1] Here, the original defined information elements in each container are omitted. In 802.21a, it will replace table 10 with added information elements.

IE_SEC_OPEN_AUTH	Whether the security policy allows open authentication.	BOOLEAN
IE_SEC_PASSWORD	Whether the security policy allows password based authentication.	Tbd
IE_SEC_CA	Certificate authority ID	Tbd
PoA-specific information elements		
IE_POA_AUTHENTICATOR_ADDR(*)	The L2 address of the authenticator, which serves the PoA.	LINK_ADDR
IE_PoA_PoS_IP_ADDR(*)	PoS's IP address; this PoS is the serving PoS of the PoA	LINK_ADDR
IE_SEC_AUTH_PROTOCOL	Which authentication protocol is supported for access authentication	Tbd
IE_SEC_EAP_METHODS	If it is EAP, then which EAP methods are supported	Tbd
IE_SEC_EAP_REAUTH	Whether to support re-authentication	BOOLEAN
IE_SEC_EAP_PREAUTH	Whether to support pre-authentication	BOOLEAN
PoA-specific higher layer service information elements		
IE_POA_AUTHENTICATOR_IP_ADDR(*)	The IP address of the authenticator which serves the PoA.	IP_ADDR
IE_PoA_PoS_IP_ADDR(*)	PoS's IP address; this PoS is the serving PoS of the PoA	IP_ADDR
IE_SEC_MOBIKE	Whether to support MOBIKE	BOOLEAN

(The IEs with (*) are currently confirmed their use.)

6.5.4 Information elements

Option 2: Change the following text as indicated:

The Information Service elements are classified into the following three groups:

a) General Information and Access Network Specific Information: These information elements give a general overview of the different networks providing coverage within an area. For example, a list of available networks and their associated operators, roaming agreements between different operators, cost of connecting to the network and network security and quality of service capabilities.
b) PoA Specific Information: These information elements provide information about different PoAs for each of the available access networks. These IEs include PoA addressing information, PoA location, data rates supported, the type of PHY and MAC layers and any

channel parameters to optimize link-layer connectivity. This also includes higher layer services and individual capabilities of different PoAs.

c) Other information that is access network specific, service specific, or vendor/network specific.

d) Security information: These information elements provide information on authentication protocols, authenticator information associated with a specific PoA, fast authentication options, EAP methods, security policies, and other security related information.

Option 2: Insert the following Information Elements in Table 10:

Name of information element	Description	Data type
Security information elements		
IE_POA_AUTHENTICATOR_ADDR(*)	The L2 address of the authenticator, which serves the PoA.	LINK_ADDR
IE_PoA_PoS_IP_ADDR(*)	PoS's IP address; this PoS is the serving PoS of the PoA.	LINK_ADDR
IE_POA_AUTHENTICATOR_IP_ADDR(*)	The IP address of the authenticator which serves the PoA.	IP_ADDR
IE_PoA_PoS_IP_ADDR(*)	PoS's IP address; this PoS is the serving PoS of the PoA.	IP_ADDR
IE_SEC_OPEN_AUTH	Whether the security policy allows open authentication.	BOOLEAN
IE_SEC_PASSWORD	Whether the security policy allows password based authentication.	Tbd
IE_SEC_CA	Certificate authority ID	Tbd
IE_SEC_AUTH_PROTOCOL	Which authentication protocol is supported for access authentication	Tbd
IE_SEC_EAP_METHODS	If it is EAP, then which EAP methods are supported	Tbd
IE_SEC_EAP_REAUTH	Whether to support re-authentication	BOOLEAN
IE_SEC_EAP_PREAUTH	Whether to support pre-authentication	BOOLEAN
IE_SEC_MOBIKE	Whether to support MOBIKE	BOOLEAN

6.5.6.2.1 IE Containers

Insert the following bullet and table at the end of 6.5.6.2.1:

- **IE_CONTAINER_SEC** – contains security related information as shown in Table 15.

Table 15 – IE_CONTAINER_SEC definition

Information element ID = (see Table G.1)	**Length** = *variable*
IE_SEC_OPEN_AUTH	
IE_SEC_PASSWORD	
IE_SEC_CA	
IE_SEC_AUTH_PROTOCOL	
IE_SEC_EAP_METHODS	
IE_SEC_EAP_REAUTH	
IE_SEC_EAP_PREAUTH	
IE_SEC_MOBIKE	

Editor's note: (1) For specifying security related IEs, two options are temporarily included. A decision is needed. (2) Some of the IEs are just place holders. More IEs may be identified.

7 Service access point (SAP) and primitives

Insert the following sub-clauses:

7.4.1.1 MIH_Capabilty_Discover.request

7.4.1.1.2 Semantics of service primitive

Parameters:

Insert the following parameter:

Name	Data type	Description
SupportedSecurityCapList	MIH_SEC_CAP	List of supported MIH security capabilities on the local MIHF.

7.4.1.2 MIH_Capabilty_Discover.indication

7.4.1.2.2 Semantics of service primitive

Parameters:

Insert the following parameter:

Name	Data type	Description

SupportedSecurityCapList	MIH_SEC_CAP	List of supported MIH security capabilities on the remote MIHF.

7.4.1.4 MIH_Capabilty_Discover.confirm

7.4.1.4.2 Semantics of service primitive

Parameters:

Insert the following parameter:

Name	Data type	Description
SupportedSecurityCapList	MIH_SEC_CAP	List of supported MIH security capabilities on the remote MIHF.

7.4.17.2 MIH_Net_HO_Candidate_Query.request

7.4.17.2.2 Semantics of service primitive

Change the following text as indicated:

MIH_Net_Ho_Candidate_Query.request(
DestinationIdentifier,
SuggestedNewLinkList,
SuggestedNewLinkCandidateAuthenticatorList,
QueryResourceReportFlag
)

Insert the following parameter:

Name	Data type	Description
SuggestedNewLinkCandidateAuthenticatorList	tbd	A list of media specific authenticator's address for the suggested candidate PoAs

7.4.17.4 MIH_Net_HO_Candidate_Query.response

7.4.17.2.2 Semantics of service primitive

Change the following text as indicated:

MIH_Net_HO_Candidate_Query.reponse(
DestinationIdentifier,
Status,
SourceLinkIdentifier,
HandoverStatus,
PerferredLinkList,
PreferedCandidateAuthenticator,
)

Insert the following parameter:

Name	Data type	Description
PreferedCandidateAuthenticator	tbd	The corresponding media specific authenticator's address of the prefered candidate PoAs

7.4.18.1 MIH_MN_HO_Candidate_Query.request

7.4.18.1.2 Semantics of service primitive

Change the following text as indicated:

MIH_MN_HO_Candidate_Query.request(
DestinationIdentifier,
SourceLinkIdentifier,
CandidateLinkList,
QoSResourceRequirments,
IPConfigurationMethods,
FA Address,
AccessRouterAddress,
PreAuthenticationFlg
)

Insert the following parameter:

Name	Data type	Description
PreAuthenticationFlg	BOOLEAN	If this value is TURE, it expects that the MIH_MN_HO_Candidate_Query.response returns the corresponding candidate media specific authenticotor's address of the prefered candidate PoAs.

7.4.18.3 MIH_MN_HO_Candidate_Query.response

7.4.18.3.2 Semantics of service primitive

Change the following text as indicated:

MIH_MN_HO_Candidate_Query.response(
DestinationIdentifier,
Status,
SourceLinkIdentifier,
PreferedCandidateLinkList,
PreferedCandidateAuthenticator
)

Insert the following parameter:

Name	Data type	Description
PreferedCandidateAuthenticator	tbd	The corresponding media specific authenticator's address of the prefered candidate PoAs

Insert the following clauses:

7.6 MIH_AUTH_SAP primitives

This clause defines primitives for MIH service access authentication and media specific proactive authentication.

7.6.1 MIH_Auth

This clause defines primitives for MIH service access authentication.

7.6.1.1 MIH_Start_Auth.request

7.6.1.1.1 Function

This primitive is used by the MIHF (MN side) to initiate the authentication process with a candidate PoS.

7.6.1.1.2 Semantics of service primitive

MIH_Start_Auth.request (
DestinationIdentifier,
SourceIdentifier
)

Parameters:

Name	Data type	Description
DestinationIdentifier	MIHF_ID	This identifies the local MIHF or a remote MIHF that will be the destination of this request.
SourceIdentifier	MIHF_ID	This identifies the invoker of this primitive, which is a remote MIHF.

7.6.1.1.3 When generated

This primitive is generated by the MN MIHF in order to start the authentication process to authenticate with the candidate PoS. This primitive is received by a remote MIHF.

7.6.1.1.4 Effect on receipt

The remote MIHF must generate a corresponding MIH_Auth_Neg request message which starts the negotiation of the authentication parameters needed.

7.6.1.2 MIH_Auth.request

7.6.1.2.1 Function

This primitive is used to perform the authentication process.

7.6.1.2.2 Semantics of service primitive

MIH_Auth.request (
DestinationIdentifier,
SourceIdentifier,
Session,
Nonce,
AuthenticationInformation
)

Parameters:

Name	Data type	Description
DestinationIdentifier	MIHF_ID	This identifies the local MIHF or a remote MIHF that will be the destination of this request.
SourceIdentifier	MIHF_ID	This identifies the invoker of this primitive, which is a remote MIHF.
Session	SESSION_ID	This identifies a security session.
Nonce	NONCE_VALUE	Represent a random value.
AuthenticationInformation	AUTH_INF_VALUE	The authentication information used to authenticate a MN.

*Only included the first time MIH_Auth.request is sent.

7.6.1.2.3 When generated

This primitive is generated after receive a MIH_Auth_Neg response or a MIH_Auth indication. This primitive requests the required information to authenticate a MN.

7.6.1.2.4 Effect on receipt

The MN must generate a MIH_Auth response in order to provide the required information if it is required.

7.6.1.3 MIH_Auth.indication

7.6.1.3.1 Function

This primitive is used by an MIHF to indicate to an MIH user that an MIH_Auth request or a MIH_Auth response was received from a remote MIHF.

7.6.1.3.2 Semantics of service primitive

MIH_Auth.indication (
AuthenticationInformation
)

Parameter:

Name	Data type	Description
AuthenticationInformation	AUTH_INF_VALUE	The authentication information used to authenticate a MN.

7.6.1.3.3 When generated

This primitive is generated after receive a MIH_Auth response. This primitive requests the required information to authenticate a MN.

7.6.1.3.4 Effect on receipt

An MIH user receiving this indication shall invoke an MIH_Auth response in order to provide authentication information.

7.6.1.4 MIH_Auth.response

7.6.1.4.1 Function

This primitive is used to perform the authentication process.

7.6.1.4.2 Semantics of service primitive

MIH_Auth.response (
DestinationIdentifier,

SourceIdentifier,
Session,
Nonce,
AuthenticationInformation
)

Parameters:

Name	Data type	Description
DestinationIdentifier	MIHF_ID	This identifies the local MIHF or a remote MIHF that will be the destination of this request.
SourceIdentifier	MIHF_ID	This identifies the invoker of this primitive, which is a remote MIHF.
Session	SESSION_ID	This identifies a security session.
*Nonce	NONCE_VALUE	Represent a random value.
AuthenticationInformation	AUTH_INF_VALUE	The authentication information used to authenticate a MN.

*Only included the first time MIH_Auth.response is sent.

7.6.1.4.3 When generated

This primitive is generated after receive a MIH_Auth indication if it is required. This primitive provides the required information to authenticate a MN.

7.6.1.4.4 Effect on receipt

The MN must generate a MIH_Auth request in order to request required information.

7.6.1.5 MIH_Finish_Auth.request

7.6.1.5.1 Function

This primitive is used to request the finalization of the established security session.

7.6.1.5.2 Semantics of service primitive

MIH_Finish_Auth.request (
DestinationIdentifier,
SourceIdentifier,
Session,
IntegrityAuth
):

Parameters:

Name	Data type	Description
DestinationIdentifier	MIHF_ID	This identifies the local MIHF or a

		remote MIHF that will be the destination of this request.
SourceIdentifier	MIHF_ID	This identifies the invoker of this primitive, which is a remote MIHF.
Session	SESSION_ID	This identifies a security session.
IntegrityAuth	INTREGRITY_DATA	This contains the message integrity data.

7.6.1.5.3 When generated

This primitive is generated by a MIHF to finalize the current security session.

7.6.1.5.4 Effect on receipt

The MIHF must generate a MIH_Auth confirm in order to confirm the session finalization.

7.6.1.6 MIH_Finish_Auth.response

7.6.1.6.1 Function

This primitive is used to confirm from the remote MIHF that the authentication session has been closed.

7.6.1.6.2 Semantics of service primitive

MIH_Finish_Auth.response (
DestinationIdentifier,
SourceIdentifier,
Session,
IntegrityAuth
)

Parameters:

Name	Data type	Description
DestinationIdentifier	MIHF_ID	This identifies the local MIHF or a remote MIHF that will be the destination of this request.
SourceIdentifier	MIHF_ID	This identifies the invoker of this primitive, which is a remote MIHF.
Session	SESSION_ID	This identifies a security session.
IntegrityAuth	INTREGRITY_DATA	This contains the message integrity data.

7.6.1.6.3 When generated

This primitive is generated by a remote MIHF after receiving a MIH_Finish_Auth request.

7.6.1.6.4 Effect on receipt

The MIHF must generate a MIH_Auth confirm in order to confirm the session finalization.

7.6.1.7 MIH_Finish_Auth.confirm

7.6.1.7.1 Function

This primitive is used to confirm the finalization of the established security session.

7.6.1.7.2 Semantics of service primitive

MIH_Finish_Auth.confirm (
SourceIdentifier,
Session
)

Parameters:

Name	Data type	Description
SourceIdentifier	MIHF_ID	This identifies the invoker of this primitive, which is a remote MIHF.
Session	SESSION_ID	This identifies a security session.

7.6.1.7.3 When generated

This primitive is generated by a MIHF to finalize the current security session.

7.6.1.7.4 Effect on receipt

The security session finishes.

7.6.2 MIH_Pro_Auth

This clause defines primitives for MIH assisted proactive authentication.

7.6.2.1 MIH_Pro_Auth.start

7.6.2.1.1 Function

This primitive is to indicate starting proactive authentication

7.6.2.1.2 Semantics of service primitive

MIH_Pro_Auth.start (

)

Parameters:

Name	Data type	Description

7.6.1.1.3 When generated

7.6.1.1.4 Effect on receipt

7.6.2.2 MIH_Pro_Auth.request

7.6.2.2.1 Function

7.6.2.2.2 Semantics of service primitive

MIH_Auth.request (

)

Parameters:

Name	Data type	Description

7.6.1.2.3 When generated

7.6.1.2.4 Effect on receipt

7.6.2.3 MIH_Pro_Auth.response

7.6.2.3.1 Function

This primitive is used by an MIHF to indicate to an MIH user that an MIH_Auth request or a MIH_Auth response was received from a remote MIHF.

7.6.2.3.2 Semantics of service primitive

MIH_Auth.indication (

)

Parameter:

Name	Data type	Description

7.6.2.3.3 When generated

7.6.2.3.4 Effect on receipt

7.6.3 MIH_Key

This clause specifies primitives for triggering key distribution.

7.6.3.1 MIH_Push_Key.request

7.6.3.1.1 Function

This primitive is used to request to a remote MIHF (PoS) to install a key in a target PoA.

7.6.3.1.2 Semantics of service primitive

MIH_Push_Key.request (
DestinationIdentifier,
SourceIdentifier,
Session,
PoAIndentifier,
IntegrityAuth
)

Parameters:

Name	Data type	Description
DestinationIdentifier	MIHF_ID	This identifies the local MIHF or a remote MIHF that will be the destination of this request.
SourceIdentifier	MIHF_ID	This identifies the invoker of this primitive, which is a remote MIHF.
Session	SESSION_ID	This identifies a security session.
PoAIdentifier	PoA_ID	This identifies a PoA.
IntegrityAuth	INTREGRITY_DATA	This contains the message integrity data.

7.6.3.1.3 When generated

This primitive is generated by an MIHF from the MN to request a remote MIHF, the Serving PoS, to install a key in a target PoA.

7.6.3.1.4 Effect on receipt

The remote MIHF must generate a MIH_Push_Key response in order to confirm that the key has been installed successful.

7.6.3.2 MIH_Push_Key.indication

7.6.3.2.1 Function

This primitive is used to send a key to the corresponding MIH User by a remote MIHF (PoS side).

7.6.3.2.2 Semantics of service primitive

MIH_Push_Key.indication (
MS-Key
)

Parameter:

Name	Data type	Description
MS-Key	KEY	Identifies a media specific key

7.6.3.2.3 When generated

This primitive is generated by a remote MIHF after receiving a MIH_Push_Key response to send a key to the corresponding MIH User.

7.6.3.2.4 Effect on receipt

A media specific key is delivered to the corresponding MIH User.

7.6.3.3. MIH_Push_Key.response

7.6.3.3.1 Function

This primitive is used to confirm that the key installation has been carried out.

7.6. 3.3.2 Semantics of service primitive

MIH_Push_Key.response (
DestinationIdentifier,
SourceIdentifier,
Session,
PoAIndentifier,
IntegrityAuth
)

Parameters:

Name	Data type	Description
DestinationIdentifier	MIHF_ID	This identifies the local MIHF or a

		remote MIHF that will be the destination of this request.
SourceIdentifier	MIHF_ID	This identifies the invoker of this primitive, which is a remote MIHF.
Session	SESSION_ID	This identifies a security session.
IntegrityAuth	INTREGRITY_DATA	This contains the message integrity data.

7.6.3.3.3 When generated

This primitive is generated by a remote MIHF after receiving a MIH_Push_Key request to confirm that a key installation has been performed.

7.6.3.3.4 Effect on receipt

The remote MIHF must generate a MIH_Push_Key confirm in order to confirm that the key has been installed successful.

7.6.3.4 MIH_Proact_Pull_key.request

7.6.3.4.1 Function

This primitive is used by the MN in order to request a key installation in a target PoA.

7.6.3.4.2 Semantics of service primitive

ParMIH_Proact_Pull_key.request (
DestinationIdentifier,
SourceIdentifier,
Session,
PoAIndentifier,
ProactivePullInformation,
IntegrityAuth
)

Parameters:

Name	Data type	Description
DestinationIdentifier	MIHF_ID	This identifies the local MIHF or a remote MIHF that will be the destination of this request.
SourceIdentifier	MIHF_ID	This identifies the invoker of this primitive, which is a remote MIHF.
Session	SESSION_ID	This identifies a security session.
PoAIdentifier	PoA_ID	This identifies a PoA.
ProactivePullInformation	PROACT_PULL_INF	This contains specific data to carry out the key installation.
IntegrityAuth	INTREGRITY_DATA	This contains the message integrity

		data.

7.6.3.4.3 When generated

This primitive is generated by the MN in order to perform a key installation in a target PoA through the Serving PoS.

7.6.3.4.4 Effect on receipt

The MN must generate a MIH_Proact_Pull response in order to provide the required information.

7.6.3.5 MIH_Proact_Pull_key.indication

7.6.3.5.1 Function

This primitive is used by the remote MIHF in order to notify the corresponding MIH user about the reception of a MIH_Proact_Pull request.

7.6.3.5.2 Semantics of service primitive

MIH_Proact_Pull_key.indication (
PoAIndentifier,
ProactivePullInformation,
)

Parameters:

Name	Data type	Description
PoAIdentifier	PoA_ID	This identifies a PoA.
ProactivePullInformation	PROACT_PULL_INF	This contains specific data to carry out the key installation.

7.6.3.5.3 When generated

This primitive is generated by the MN in order to perform a key installation in a target PoA through the Serving PoS.

7.6.3.5.4 Effect on receipt

The MN must generate a MIH_Proact_Pull response in order to provide the required information to perform the key distribution.

7.6.3.6 MIH_Proact_Pull_key.response

7.6.3.6.1 Function

This primitive is used by the Serving PoS in order to carry out the key distribution.

7.6.3.6.2 Semantics of service primitive

MIH_Proact_Pull_key.response (
DestinationIdentifier,
SourceIdentifier,
Session,
PoAIndentifier,
ProactivePullInformation,
IntegrityAuth
)

Parameters:

Name	Data type	Description
DestinationIdentifier	MIHF_ID	This identifies the local MIHF or a remote MIHF that will be the destination of this request.
SourceIdentifier	MIHF_ID	This identifies the invoker of this primitive, which is a remote MIHF.
Session	SESSION_ID	This identifies a security session.
ProactivePullInformation	PROACT_PULL_INF	This contains specific data to carry out the key installation.
IntegrityAuth	INTREGRITY_DATA	This contains the message integrity data.

7.6.3.6.3 When generated

This primitive is generated after receive a MIH_Proact_Pull indication. This primitive provide the required information to perform the key distribution.

7.6.3.6.4 Effect on receipt

The MN must generate a MIH_Proact_Pull request in order to provide the required information until the key distribution is completed.

8 Media independent hand over protocols

Insert the following sub-clauses:

8.4.3 Protected Message Format

8.4.3.1 MIH Security PDU

An MIH Security (MIHS) PDU is an MIH PDU that has an MIHS header, followed by followed by optional Source and Destination MIHF-ID TLVs, followed by an optional Session ID TLV, followed by a TLS TLV. An MIHS header is an MIH protocol header containing the following information.

- Version: the version of MIH protocol
- Ack-Req: 0
- Ack-Rsp: 0
- UIR: 0
- M:0
- FN:0
- SID: 5 (Security Service)
- Opcode: 2 (Indication)
- TID: 0

A Session ID TLV is associated with the pair of MIHFs associated with the MIH SA. Therefore, Source and Destination MIHF Identifier TLVs do not need to be carried in an MIHS PDU in existence of an MIH SA, and a Session ID TLV is carried instead. Source and Destination MIHF Identifier TLVs are carried in a MIHS PDU in absence of an MIH SA or when the sender's transport address has been changed. In the latter case, the mapping between the sender's transport address and the MIHF identifier shall be updated, and an MIH Registration request or response message may be contained in the TLS TLV.
The structure of MIHS PDU during TLS handshake is shown in **Figure xxx**. The structure of MIHS PDU in existence of an MIH SA is shown in **Figure**. The structure of MIHS PDU upon Transport Address Change is shown in **Figure** .

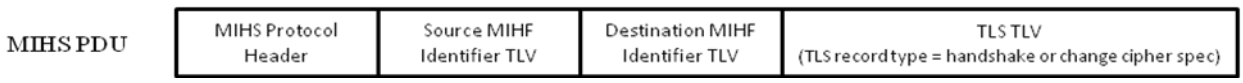


Figure xxx: MIHS PDU during TLS handshake

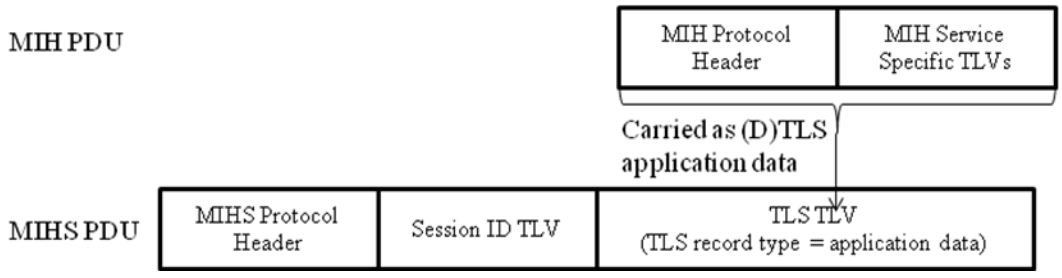


Figure xxx: MIHS PDU in existence of MIH SA

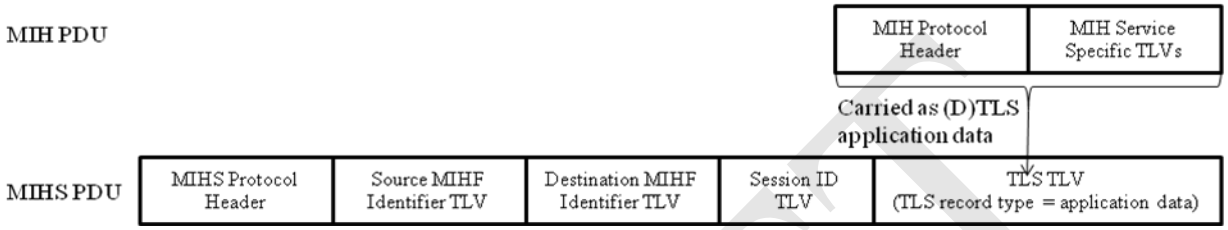


Figure xxx: MIHS PDU upon Transport Address Change

(Editor's note: The Figures will numbered.)

8.6 MIH protocol messages

Insert the following subclause:

8.6.X.1 MIH_Security Indication

MIH Header Fields (SID=5, Opcode=2, AID-xx)
Source Identifier = sending MIHF ID (optional) (Source MIHF ID TLV)
Destination Identifier = receiving MIHF ID (optional) (Destination MIHF ID TLV)
Session Identifier = session id (optional) (Session ID TLV)
TLS = transport layer security (TLS TLV)

8.6.X.2 MIH_Start_Auth.request

MIH Header Fields (SID = 2, Opcode = 1, AID = XX-1)
Destination Identifier = receiving MIHF ID (Destination MIHF ID TLV)
Source Identifier = sending MIHF ID

(Source MIHF ID TLV)

8.6.X.3 MIH_Auth.response

MIH Header Fields (SID = 2, Opcode = 1, AID = XX-1)
Destination Identifier = receiving MIHF ID (Destination MIHF ID TLV)
Source Identifier = sending MIHF ID (Source MIHF ID TLV)
Session (Session TLV)
Nonce (Nonce TLV)
AuthenticationInformation (Authentication Information TLV)

8.6.X.4 MIH_Auth.indication

MIH Header Fields (SID = 2, Opcode = 3, AID = XX-2)
AuthenticationInformation (Authentication Information TLV)

8.6.X.5 MIH_Auth.response

MIH Header Fields (SID = 2, Opcode = 3, AID = XX-3)
Destination Identifier = receiving MIHF ID (Destination MIHF ID TLV)
Source Identifier = sending MIHF ID (Source MIHF ID TLV)
Session (Session TLV)
Nonce (Nonce TLV)
AuthenticationInformation (Authentication Information TLV)

8.6.X.6 MIH_Finish_Auth.request

MIH Header Fields (SID = 2, Opcode = 1, AID = XX-1)
Destination Identifier = receiving MIHF ID (Destination MIHF ID TLV)

Source Identifier = sending MIHF ID (Source MIHF ID TLV)
Session (Session TLV)

8.6.X.7 MIH_Finish_Auth.response

MIH Header Fields (SID = 2, Opcode = 2, AID = XX-2)
Destination Identifier = receiving MIHF ID (Destination MIHF ID TLV)
Source Identifier = sending MIHF ID (Source MIHF ID TLV)
Session (Session TLV)

8.6.x.8 MIH_Pro_Auth.Start

MIH Header Fields (SID=3, Opcode=3, AID-xx)
Source Identifier = sending MIHF ID (Source MIHF ID TLV)
Destination Identifier = receiving MIHF ID (Destination MIHF ID TLV)
Auth_Start TLV

8.6.X.9 MIH_Pro_Auth.Request

MIH Header Fields (SID=3, Opcode=1, AID-xx)
Source Identifier = sending MIHF ID (Source MIHF ID TLV)
Destination Identifier = receiving MIHF ID (Destination MIHF ID TLV)
Proactive EAP message Frame TLV

8.6.X.10 MIH_Pro_Auth.Response

MIH Header Fields (SID=3, Opcode=2, AID-xx)
Source Identifier = sending MIHF ID (Source MIHF ID TLV)
Destination Identifier = receiving MIHF ID (Destination MIHF ID TLV)
Proactive EAP message Frame TLV

(Editor's note: Need to determine in which Clause these messages should be defined.)

8.6.X.11 MIH_Push_Key.request

MIH Header Fields (SID = 2, Opcode = 1, AID = XX-1)
Destination Identifier = receiving MIHF ID (Destination MIHF ID TLV)
Source Identifier = sending MIHF ID (Source MIHF ID TLV)
Session (Session TLV)
PoAIdentifier (PoA Identifier TLV)
IntegrityAuth (Integrity Auth TLV)

8.6.X.12 MIH_Push_Key.indication

MIH Header Fields (SID = 2, Opcode = 3, AID = XX-2)
MS-Key (Media Specific Key TLV)

8.6.X.13 MIH_Push_Key.response

MIH Header Fields (SID = 2, Opcode = 2, AID = XX-3)
Destination Identifier = receiving MIHF ID (Destination MIHF ID TLV)
Source Identifier = sending MIHF ID (Source MIHF ID TLV)
Session (Session TLV)
IntegrityAuth (Integrity Auth TLV)

8.6.X.14 MIH_Proact_Pull_key.request

MIH Header Fields (SID = 2, Opcode = 1, AID = XX-1)
Destination Identifier = receiving MIHF ID (Destination MIHF ID TLV)
Source Identifier = sending MIHF ID

(Source MIHF ID TLV)
Session (Session TLV)
PoAIdentifier (PoA Identifier TLV)
ProactivePullInformation (Proactive Pull Information TLV)
IntegrityAuth (Integrity Auth TLV)

8.6.X.15 MIH_Proact_Pull_key.response

MIH Header Fields (SID = 2, Opcode = 2, AID = XX-1)
Destination Identifier = receiving MIHF ID (Destination MIHF ID TLV)
Source Identifier = sending MIHF ID (Source MIHF ID TLV)
Session (Session TLV)
ProactivePullInformation (Proactive Pull Information TLV)
IntegrityAuth (Integrity Auth TLV)

9 MIH protocol protections

This clause specifies options and mechanisms to protect media independent handover protocol. The MIH protocol can be protected through the transport protocols at layer 2 or layer 3. It can also be protected through MIH specific mechanisms. To apply MIH specific protection mechanisms, a mobile node and a point of service need to establish cryptographic keys. We will first discuss protections through transport protocols. Then we will specify MIH specific key establishment mechanisms. The protected MIH message format is introduced after key establishment.

9.1 Protections through transport protocols

MIH messages can be carried over wireless protocols in layer 2 such as defined in IEEE 802.11u or layer 3 as defined in IETF RFC 5677. In the following we will discuss the security protections provided through the transport protocol and identify issues with each protection mechanism.

9.1.1 Protection through layer 2

When MIH messages are transported over a layer 2 protocol, the protections may be provided through the layer 2 protocol such as TKIP and CCMP specified in IEEE 802.11.

The protections in layer 2 are usually established with L2 identifiers for an MN and a PoS such as MAC address. MIH messages are protected together with other data. Furthermore, if MIH messages are transported over different layer 2 protocols, then the protections may be different.

On the other hand, such protections through a layer 2 protocol will not require any change either on MIH protocol or layer 2 protocol.

9.1.2 Protection through IPsec

When MIH messages are transported over IP as defined in RFC 5677, it may be protected by IPsec. When IPv6 is implemented in a mobile node and a PoS, then IPsec is mandatory. In this case, MIH messages are protected at IP layer as an IP payload in each IPsec packet.
For a pair of IP nodes with fixed IP addresses, the IPsec Security Associations (SAs) are established through Internet Key Exchange (IKEv1 or IKEv2). However, in case of MIH message protection, the IP address of a mobile node may be dynamic. In this case, a protocol suite defined by RFC 4555 - "IKEv2 Mobility and Multihoming Protocol (MOBIKE)" may be used to establish SAs between an MN and a PoS (aka MoS as defined in RFC 5677).

It is similar to IKEv2, MOBIKE is a heavy weight protocol. The MOBIKE RFCs are explicitly defined for tunnel-mode IPSec connections. In several places of RFC 4555, they say that transport-mode could be used, but it would require new RFCs to define the use of that mechanism within MOBIKE.

IPsec protocols are well defined and can provide proper protections for its IP payload. When SAs are established between an MN and a PoS, it can provide end-to-end protections. Using IPsec will not require any changes to either MIH protocol or IPsec.

Similar to layer 2 protection, the protections through IPsec are not MIH specific. However, for the mutual authentication through MOBIKE, the certificates may be issued on identifiers which are related to MIH applications. From this point of view, IPsec is closer to MIH specific protection, compared to L2 protection.

(Editor's note: MIH may be protected through TLS, if a new port is assigned in IETF and proper modification to TLS is defined. This option is not included in this draft. If during the development of this standard, certain IETF standards will coordinate these needs, then (D)TLS may be re-considered for MIH protections.)

9.2 Protections through MIH specific mechanisms

In order to establish MIH specific protections for MIH protocol between a MN and a PoS, it needs to execute a key establishment protocol first. In this standard, we introduce the following two options for key establishment.

9.2.1 Key establishment through (D) TLS

In this option, a mobile node and a PoS execute a TLS [RFC5246] or DTLS [RFC4347] to establish MIH security associations, where the MN is TLS client, while the PoS is the TLS server. The mutual authentication is executed through either pre-shared secret key or a public key certificate issued by a trusted third party such as a certificate authority. In this option, the authentication may or may not be related to an access control. It can be an access authentication for MIH service, if a PoS can hold service credentials for the mobile nodes. The (D)TLS handshake and protected MIH messages are illustrated in Figure xx.

(Editor's note: Here the access controller can be applied. Need to discuss with proposers.)

Figure xx. (D)TLS handshake and Protected MIH Messages

TLS handshake is carried over MIH protocol and MIH SAs are established between two MIHF peers. The primitives for handling (D)TLS are defined in 7.4 and 7.6. The MIH messages to carry (D)TLS handshake messages are specified in Clause 8.6.

(Editor's note: It needs to specify how to use the output keys from (D)TLS.)

9.2.2 Key establishment through an MIH access authentication

If MIH service is subscription based and provided by a service provide, then a MIH service access authentication may be needed to authorize the service to a mobile node. In this case, an access authentication may include a key establishment subroutine to allow a PoS to obtain a master session key so that MIH security associations can be established through the master session key between the MN and the PoS.

9.2.2.1 MIH service access authentication

This clause describes MIH service access authentication process. The primitives are specified in Clause 7.6.1.

In this standard, it is assumed that EAP [RFC3748] or EAP Re-authentication (ERP) [RFC5296] is used as an authentication protocol with an MN as the peer and a PoS as the authenticator. An EAP server may be used as a backend server.

For the interface between a MN and a PoS, MIH is acting as an EAP lower layer. That is, at the MN, an EAP message is generated at an MIH user and passes to then MIHF. When it reaches the PoS, the MIHF in PoS will pass the message to the MIH user o process it. For an EAP message from the PoS to the MN, it will also be generated by the MIH user in the PoS and passes to MIHF. At the MN, the message is passed to the MIH user to process. The protocol stack is illustrated in Firgure xxx.

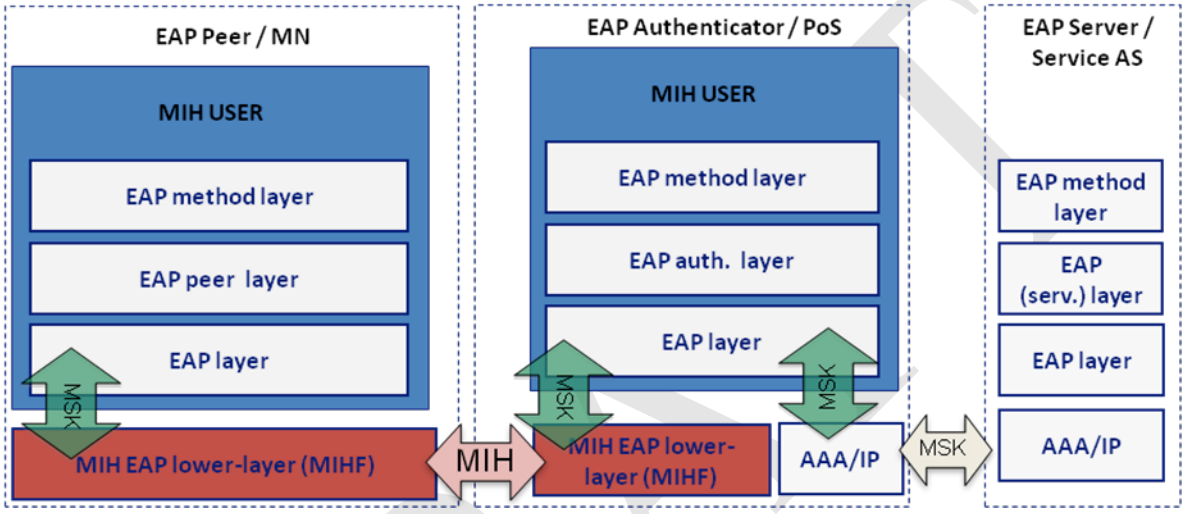


Figure xxx: Protocol stacks in each entity

The authentication is divided to the following phases:

- *Negotiation Phase*. In this phase, both the MN and the PoS exchange unprotected MIH messages in order to agree on the services, which want to be accessed by the MN. PoS provides a list of the available services and the MN chooses, for example, by priority the service or services that it supports.
- *Media Independent Service Authentication phase*. The MN authenticates against a PoS (is acting as EAP authenticator) using a Service AS. To achieve this, EAP is transported by MIH protocol to the current PoS, which manages the MN's communications. In order to carry out the authentication the PoS may need a backed authentication server (Service AS, e.g. AAA server) to verify the MN's credentials. After performing the authentication, keying material will be shared between the MN and the PoS. So, both at MN's and PoS' MIH layer keying material is exported and it could be used to protect the rest of the communication. The protection mechanisms will be specified in Clause 9.2.3. The protected message format is specified in Clause 8.4.3. In order to preserve the security of the exported keying material,

the exported MSK is used as root key to derive new session keys which are use to protect the MIH packets. How this key hierarchy is carried out is described in Clause 9.2.2.2.

- *Service Access phase*. At this point, the MN is authenticated and authorized to use the MIH services, agreed and provided by the PoS. The authentication session is already established and the MIH protocol is protected by using the key material obtained in the Media Independent Service Authentication Phase. This phase is related with Clause 9.2.2.2 for key derivation and Clause 9.2.3 for protecting MIH protocol.
- *Finalization phase*. When the MN and the PoS desire to finish the authentication session in order to release resources and the MN' state related with the provided services.

These phases are illustrated in Figure xxx. The primitives for MIH access authentication are specified in Clause 7.6.

(Editor's note: Service access phase may not be a part of authentication. Need further discussion.)

(Editor's note: The message flows may be included in Annex N.)

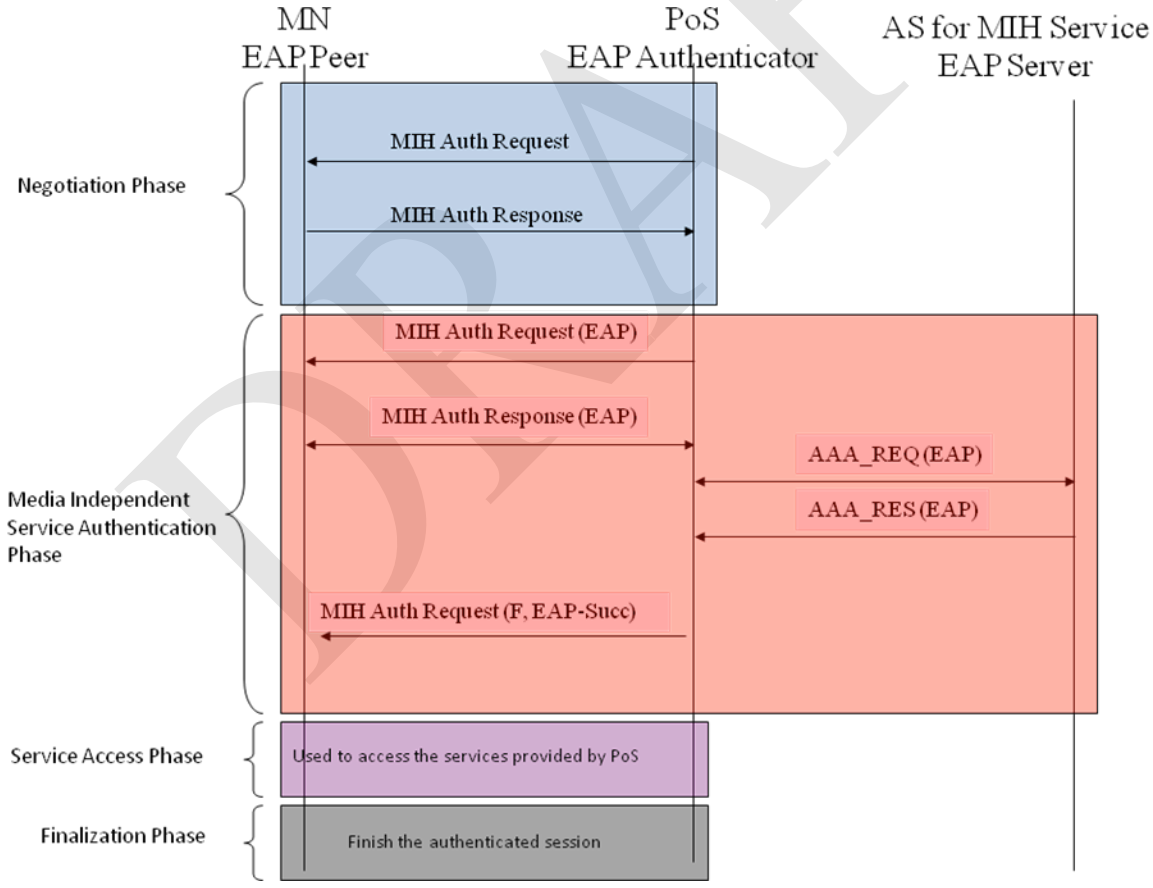


Figure xxx: MIH Service Access Authentication

9.2.2.2 Key derivation and key hierarchy

Upon a successful service access authentication, the authenticator, PoS, obtains a master session key (MSK) or a re-authentication MSK. From MSK or rMSK, a key hierarchy is derived as shown in Figure xxx.

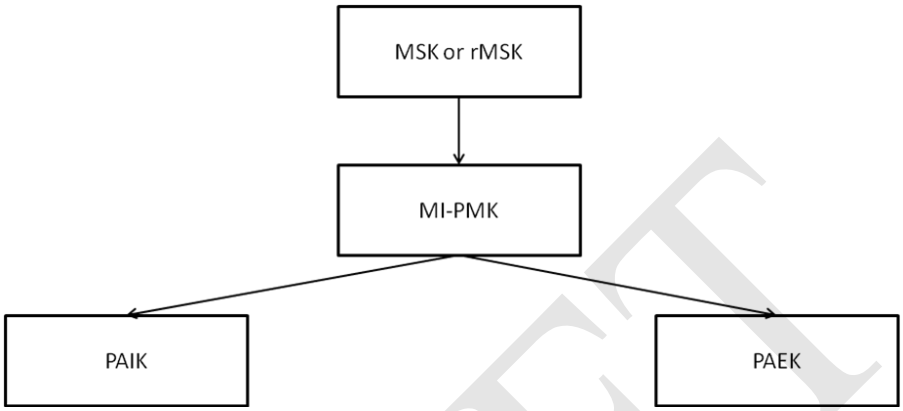


Figure xxx MIH Protection Key Hierarchy

Using the MSK or rMSK provided by the EAP authentication a MI-PMK is derived in order to preserve the key material security. Using this MI-PMK as root key, other session keys are derived:

- PAIK is used to provide integrity protection to the MIH protocol.
- PAEK is used to provide confidentiality to the MIH protocol by encrypting MIH data.

If L bits of keying material are needed, then the default Key derivation function KDF is defined as follows:

PRF+ (K,S) = T1 | T2 | T3 | T4 | ...|Tn)

Where:

'|' means concatenation
T1 = PRF (K, S | 0x01)
T2 = PRF (K, T1 | S | 0x02)
T3 = PRF (K, T2 | S | 0x03)
T4 = PRF (K, T3 | S | 0x04)
…

until L bits of keying material are generated.

The key hierarchy key hierarchy is derived in the following way:

- *MI-PMK = KDF(MK, "MI-PMK" | RAND_P | RAND_A | length)*
    - Length of MI-PMK is 64 octets

- MK (Master Key): MSK or rMSK
- RAND_P: A random number generated by peer
- RAND_A: A random number generated by authenticator

- *PAIK = KDF(MI-PMK, "integrity key" | RAND_P | RAND_A | length)*
  - Length of PAIK is 64 octets
  - MI-PMK (Master Key)
  - RAND_P: A random number generated by peer
  - RAND_A: A random number generated by authenticator
- *PAEK = KDF(MI-PMK, "encryption key" | RAND_P | RAND_A | length)*
  - Length of PAEK is 64 octets
  - MI-PMK (Master Key)
  - RAND_P: A random number generated by peer
  - RAND_A: A random number generated by authenticator

(Editor's note: (1) The key hierarchy needs to be coordinated with (D)TLS option. The session keys PAIK and PAEK are also depending on what algorithm will be selected for MIH SA. For example, if AES CCM or GCM is selected, then it is an authenticated encryption mode. Furthermore, it shall not need 64 octets of keying material in any case. (2) The messages to exchange RAND_P and RAND_A shall be specified explicitly.)

9.2.3 MIH security associations

(Editor's note: MIH SA needs to be defined.)

10 MIH assisted proactive authentication

In a handover from a service PoA to a target PoA, a mobile node needs to authenticate to the target network through a required authentication mechanism. This clause defines MIH assisted proactive authentication. The authentication protocol is initiated with the target network through a PoA as indicated in Figure xx.

This standard introduces two options to conduct the proactive authentication with the targeted network. The first option is called media specific proactive authentication. In a media specific proactive authentication, a MN conduct an authentication with the targeted network as it is

specified for that network. In this case, the authenticator is a media specific authenticator (MSA). The authentication messages are passed between the MN and the MSA through a MIH PoS. The authentication messages between the MN and PoS are carried through MIH messages. The second option is to bundle the media access proactive authentication to MIH service access authentication as specified in clause 9. In this case, at the end of MIH service access authentication, the MN and PoS also establish a key or keys for a target PoA. The key(s) are distributed from the PoS to PoA so that when the handover happens, the MN can establish a protected link with the PoA.

10.1 Media Specific Proactive Authentication

In a media specific proactive authentication, a PoS passes authentication messages between the mobile node and the media specific authentication (MSA). The protocol stacks in each interface are illustrated in Figure xxx.

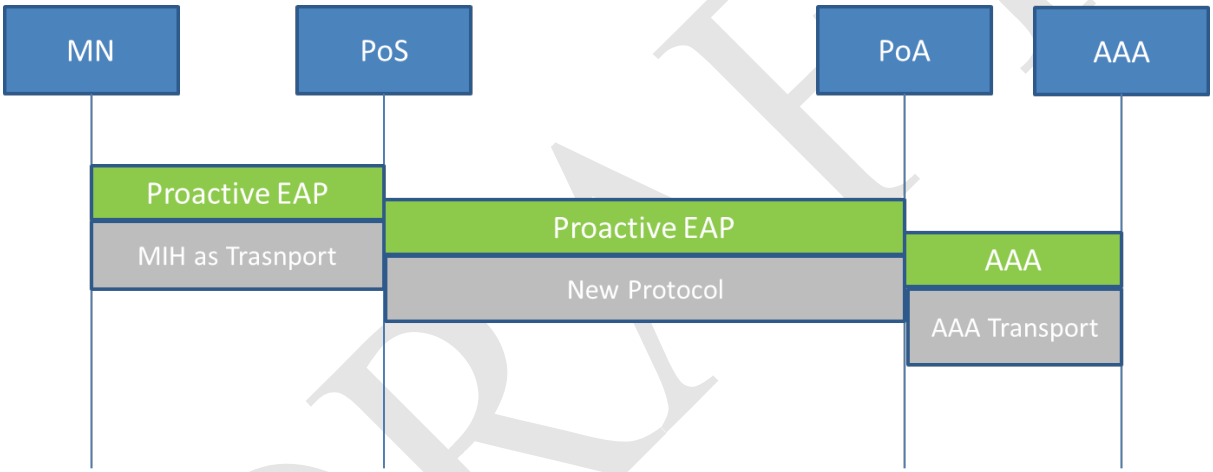


Figure xxx. Media Specific Proactive Authentication Protocol Stacks

10.1.1 Procedures of MIH assisted proactive authentication

The procedure of MIH assistant proactive authentication is consisted of the following steps:

1. POS and Candidate Media Specific Authenticator Discovery

Before MN initiates MIH assisted proactive authentication, MN needs to know the PoS's address and the candidate media specific authenticator's MAC addresses. The discovery of PoS's address has already been specified by IEEE 21 base specification. The discovery of corresponding candidate MSA's address could be done by using extended IEEE802.21 information service. The information elements (IEs) are specified in 6.5.4.

2. Perform of Proactive EAP authentication

The proactive EAP authentication or ERP message could be encapsulated in to the extended MIH messages as L2 frames. When the PoS receives the encapsulated MIH message, it decapsulates it, then forwards the EAP message to the media specific candidate authenticator. The EAP message could be encoded as OCTET_STRING, the PoS needs not to implement EAP protocol; It may simply forward the EAP messages to the MSA.

After successful proactive EAP authentication, the MN and AS derives the related keying material and candidate media specific authenticator can get the keying material from the AS using AAA protocol. The packet format of the MIH assisted EAP proactive authentication is depicted in Figure xxx.

3. Media specific association handshake

When the MN decide to handover to the candidate network, the MN and target PoA, which associated with the MSA, perform a media specific association based on the keying material derived by the proactive EAP authentication. For example, this could be 4-way handshake in 802.11.

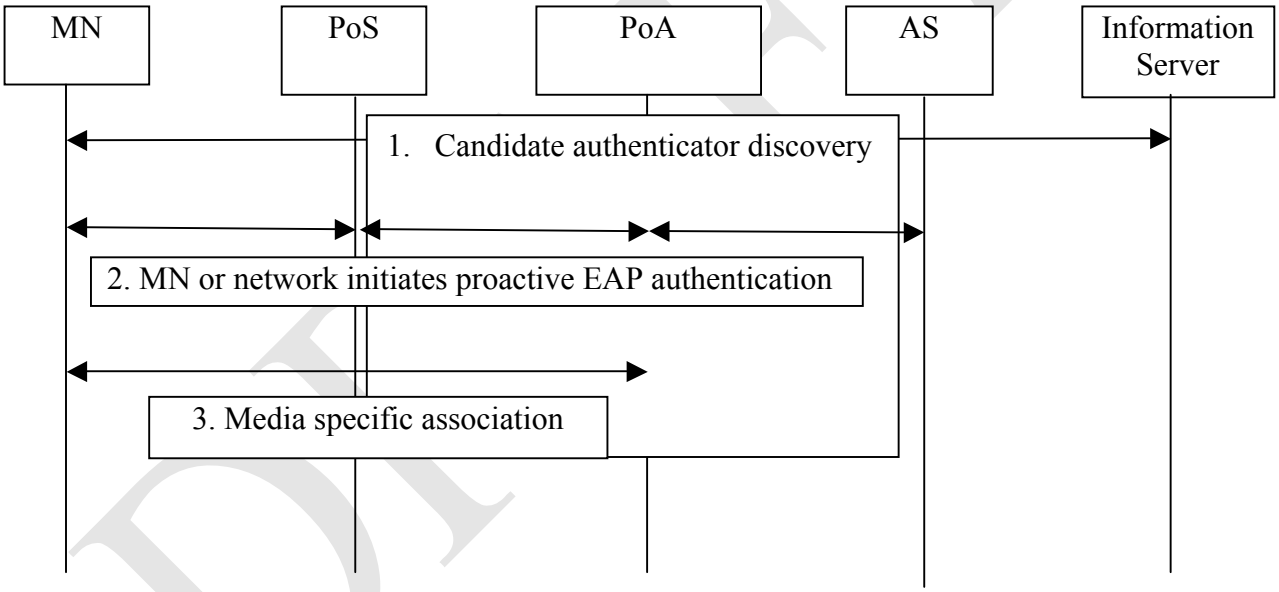


Figure xxx Procedure of MIH assisted proactive EAP authentication

10.1.2 Proactive authentication message format

When a proactive authentication is executed through EAP [RFC3748], the EAP packets are carried by MIH messages. MIH primitives are defined in Clause 7.6.2 for proactive authentication. The messages are defined in 8.6.x. The proactive authentication message format is illustrated in Figure xxx.

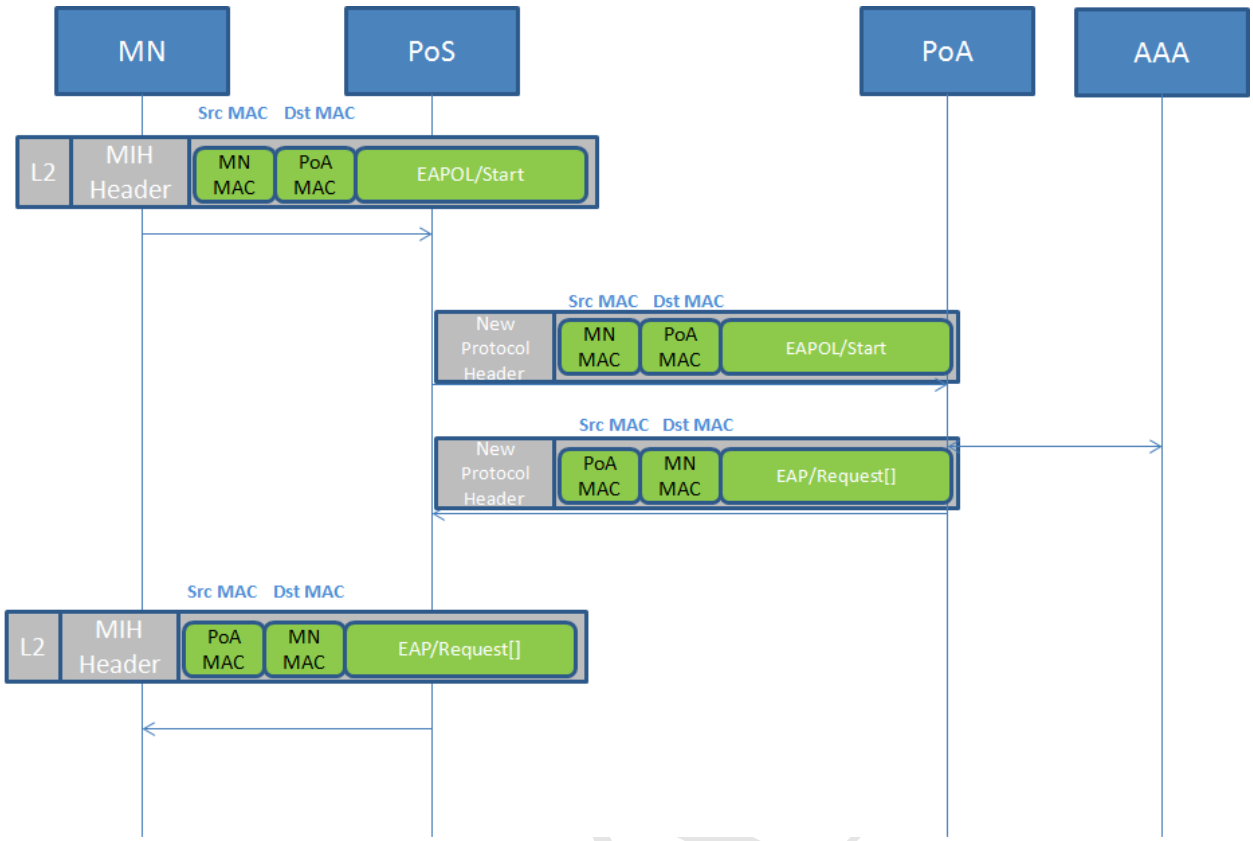
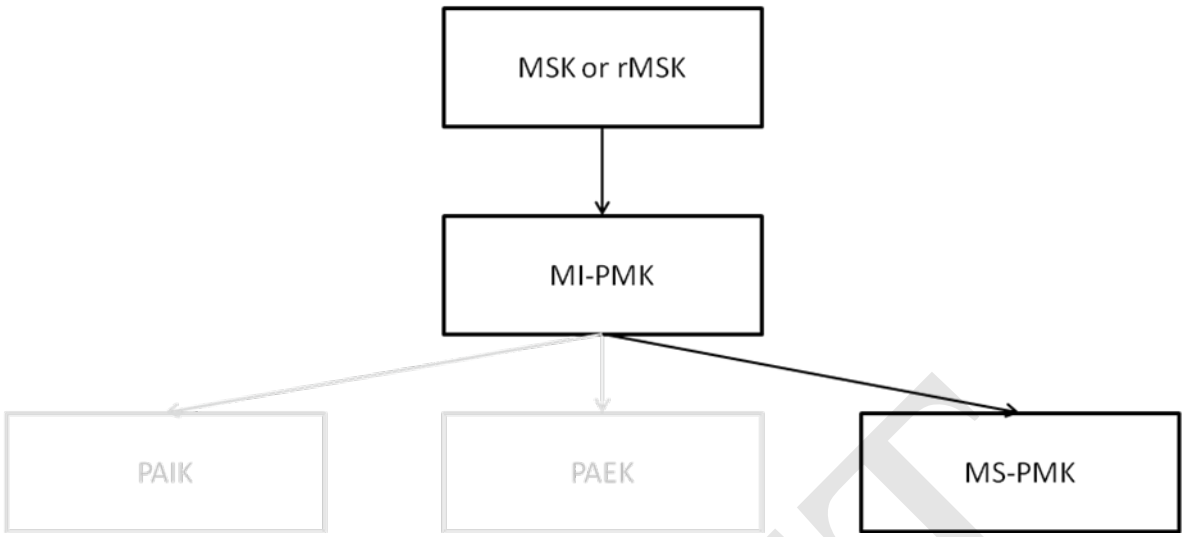


Figure xxx. MIH Assisted proactive authentication message format

10.2 Bundle to MIH Service Access Authentication

A media access authentication may be bundled with a MIH service access authentication when the trust relationship between media specific network access provider and the MIH service provider allows. In this case, at the end of a successful authentication, a PoS will derive not only keys for MIH message protection but also a key called media specific MSK (MS-MSK) to be used by a target PoA. The new key hierarchy is illustrated in Figure xxx.



The MS-PMK will be derived as follows:

MS-PMK =KDF(MI-PMK, "MS-PMK" | MN_LINK_ID | POA_LINK_ID | length)

where

- Length of MS_PMK depends on each media. In the case of 802.11, Length=32.
- MN_LINK_ID: Link identifier of MN, encoded as LINK_ID data type
- POA_LINK_ID: Link identifier of media-specific authenticator, encoded as LINK_ID data type

The key MS-PMK will be distributed to a PoA where it is used as a MSK to further derive the media specific session keys to protect the media specific communication link. For example, in case of a PoA is a 802.11 AP, the MS-MSK is used to derive PMK.

The key distribution from the PoS to PoA can be done through push or pull key distribution. In general, key distribution from the PoS to PoA is out of the scope of this standard. However, MIH service can be used to trigger the key distribution. The mechanisms used to trigger the key distribution are introduced in the following sub-clauses.

(Editor's note: It must be clear that 21a does not define key distribution mechanisms. It defines mechanisms to trigger key distribution.)

10.2.1 Push key distribution

Its objective is to push a key into the target PoA by the PoS which controls that PoA. To perform the installation the MN uses the MIH protocol, which at this point could be protected, to notify the PoS to start the key installation. In the PoS, the key is pushed from MIHF to MIH user for the

further distribution to a PoA. The primitives for push key distribution are defined in 7.6.3. The message flows are illustrated in Figure xxx.

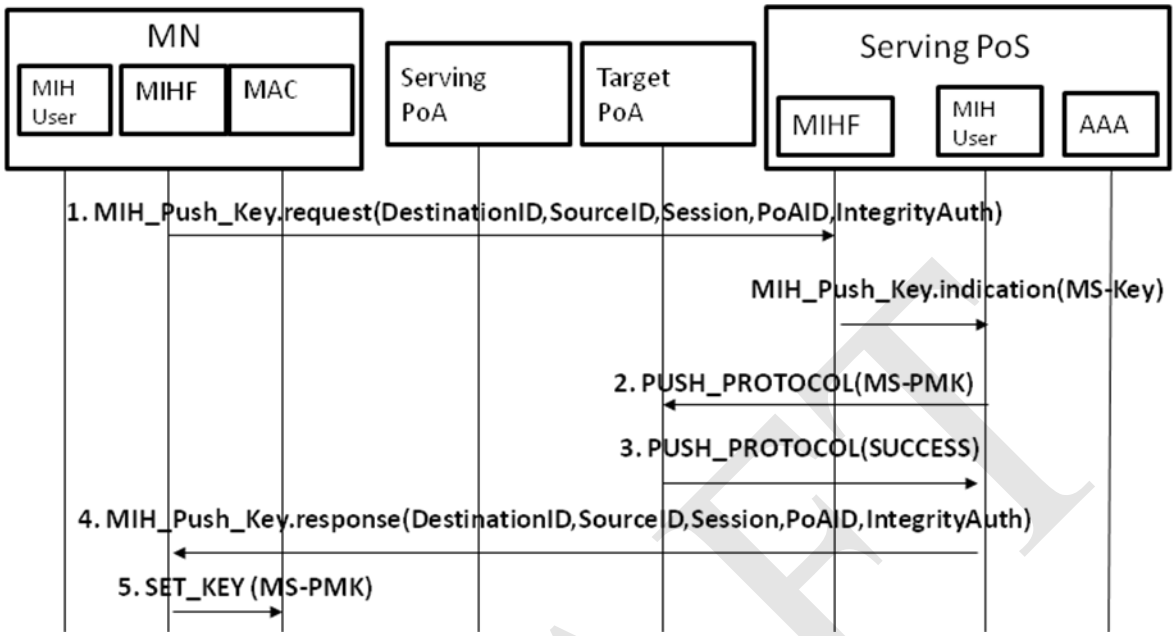


Figure xxx: Push key distribution

- PUSH_PROTOCOL: Represents the protocol used in push key distribution mechanism to install the key in the target PoA by the PoS (e.g. SNMP). (Editor's note: Is this out of the scope of 21a?)
- KEY-REQ/RES: used to request and get a key from the MIHF.
- SET_KEY: To install a key in the MAC layer.

(Editor's note: This figure is here to assist the discussion. It may be moved to a Annex N later.)

10.2.2 Reactive pull key distribution

It is performed after the MN moves to the target PoA. It assumes that both the MN and the PoS shares a symmetric key, in that way, the symmetric key is derived from the key hierarchy shared between the MN and the PoS. Therefore, the key distribution to the target PoA could be performed, for example, executing an EAP authentication where the PoS act as a local AAA server. The general message flow is depicted in Figure xxx.

(Editor's note: Is MS-PMK used as an authentication key? However, this is out of the scope of 802.21a.)

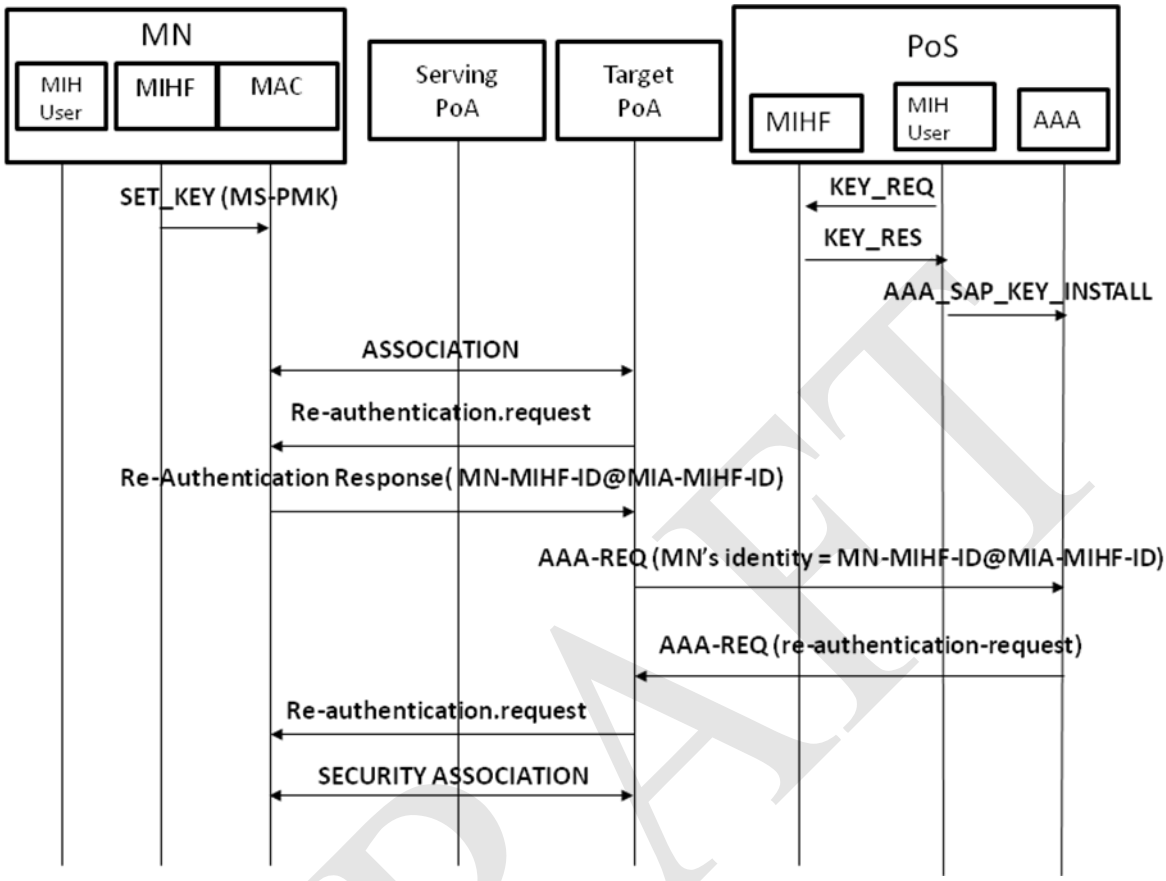


Figure xxx: Reactive pull key distribution

- KEY-REQ/RES: used to request and get a key from the MIHF.

(Editor's note: This is the only MIH message. The others are not MIH messages and not in the scope of 21.)

10.2.3 Proactive pull key distribution

This mechanism allows the MN to perform a pro-actively media specific authentication with the target PoA without being directly connected to the wireless link of the target PoA by means sending level two frames through the PoS to the target PoA. As in reactive pull key distribution, the key hierarchy shared between the MN and the PoS could be used in order to derive a shared key to be used in the authentication process, where the PoS will be acting as a local AAA.

(Editor's note: Is MS-PMK used as an authentication key? However, this is out of the scope of 802.21a.

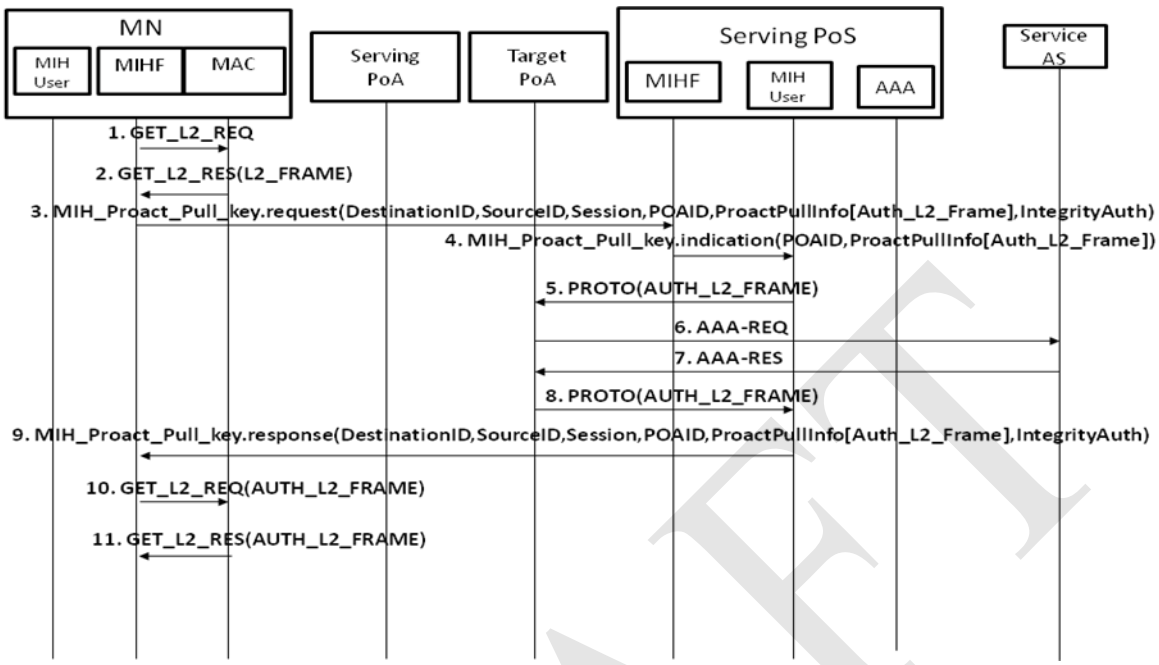


Figure xxx: Proactive pull key distribution

Annex A

(Informative)

Bibliography

Insert the following informative references:

Annex F

F.3.x Data Types for Security

Data type name	Derived from	Definition
MIH_SEC_CAP	BITMAP	

(Editor's note: Need definition from the proposers.)

Annex L

Inserting the following row to Table L.1.

Table L.1: AID Assignment

Message name	AID
MIH_Security Indication	JJ

Inserting the following rows to Table L.1.

Message Name	AID
Capability_Discover_Request	1
Capability_Discover_Response	1

Inserting the following rows to Table L.1.

Message name	Action ID
MIH_Pro_Auth_Start	TBD
MIH_Pro_auth Request	TBD
MIH_Pro_auth Response	TBD

Inserting the following TLVs to Table L.2.

Table L.2: Type values for TLV encoding

TLV type name	TLV type value	TLV data type
TLS TLV	tbd	OCTET_STRING

Table L.2: Type values for TLV encoding

TLV type name	TLV type value	TLV data type
Session ID TLV	tbd	OCTET_STRING

Table L.2: Type values for TLV encoding

TLV type name	TLV type value	TLV data type
Security capability TLV	tbd	MIH_SEC_CAP

(Editor's note: The TLV type value is to be determined.)

Annex N

(Informative)

Authentication Protocols

Editor's note: This section may include message flow for authentication protocols currently used in EAP, ERP, and AKA.